Getting Started with the 21555 Non-Transparent PCI-to-PCI Bridge

Application Note

July 2001

Order Number: 278382-001

Application Note

Contents

Figures

Tables

1.0 Introduction

This application note explains the steps required to reset, initialize, and configure the 21555 Non-Transparent PCI-to-PCI bridge (referred to as the 21555) such that it can forward memory transactions upstream and downstream through designated address windows.

The 21555 performs PCI bridging functions for non-transparent and intelligent I/O applications. The 21555 is a non-transparent PCI-to-PCI bridge that acts as a gateway to an intelligent subsystem. The 21555 functions as a bridge between two PCI processor domains, the host domain and the local domain.

Special features of the 21555 include:

- Support for independent primary and secondary PCI clocks
- Independent primary and secondary address spaces
- Address translation between the primary (host) and secondary (local) PCI buses (domains).

The 21555 creates a configuration barrier between the two PCI domains. Standard hierarchical PCI configuration methods using Type 1 configuration transactions cannot be used to access the configuration space of devices on the opposite side of the 21555. This application note serves as a guide to the initialization and configuration methodology the 21555 uses to enable independent configuration of the host and local domains.

2.0 Assumptions

This application note assumes that the 21555 is used as follows:

- A host processor resides on the primary side of the 21555
- A local processor resides on the secondary side of the 21555
- A serial ROM (SROM) is attached to the 21555 to establish initialization parameters subsequent to reset but prior to the PCI configuration process of the host and local processors
- The host processor configures all devices on the primary bus, as well as the 21555 primary PCI configuration registers
- The local processor configures all devices on the secondary bus, as well as the 21555 secondary PCI configuration registers and device specific registers
- Proper board implementation guidelines have been followed for the 21555, as described in the *21555 Non-Transparent PCI-to-PCI Bridge Hardware Implementation Application Note.*

Typically, the first time power is applied to a newly assembled 21555 subsystem, the serial ROM attached to the 21555 is blank and will need to be programmed. The *Using a Serial ROM with the 21555 Non-Transparent PCI-to-PCI Bridge Application Note* describes how to flash the serial ROM. The serial ROM registers are accessible in either memory or I/O space through the first two base address registers (BARs), respectively. These registers are accessible from either the primary or secondary interface.

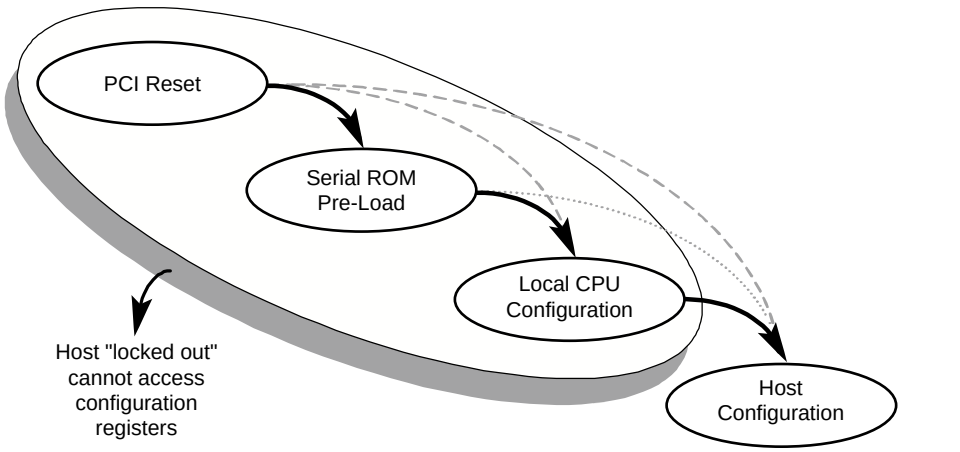
3.0 21555 Initialization

This section describes the 21555 initialization sequence. Although variations on this sequence are possible, this document limits discussion to the most typical case.

The typical 21555 configuration sequence is shown in Figure 1. The sequence consists of the following steps:

1. PCI reset (generated from host or primary bus RST#)
2. Serial ROM preload (sizes and types of BARs, as well as some 21555 device-specific configuration state, are loaded from Serial ROM prior to PCI configuration by the host and local CPUs)
3. Local CPU configuration from the secondary interface (override configuration state initialized by Serial ROM preload, set additional device-specific state, configure standard secondary PCI configuration registers, release host lockout)
4. Host CPU configuration from the primary interface (configure standard primary PCI configuration registers)

Figure 1. 21555 Initialization Sequence



The following sections describe the initialization sequence steps in more detail.

3.1 Reset

The 21555 implements five different types of reset mechanisms as described in Table 1.

The primary reset signal **p_rst_l** is an input signal and resets all chip state and data buffers, including all resetable registers. Assertion of signal **p_rst_l** is the most standard method of starting the 21555 initialization sequence.

The secondary reset signal **s_rst_l** is an output, and is asserted under all four of the reset mechanisms described in Table 1. Signal **s_rst_l** is deasserted upon reset release as noted in the table. Signal **s_rst_l** is not an open drain output and should not be connected to other potential signal drivers of the secondary PCI bus reset. If another source of secondary bus reset is desired, then **s_rst_l** should remain unconnected. The 21555 can be reset from the secondary PCI bus using the s_rst_in_l feature.

Table 1. 21555 Reset Mechanisms

Reset Event	Effect of Reset Event						Reset Release[a]
	Primary PCI State Machine	Secondary PCI State Machine	Registers	Data Buffers	s_rst_l	Serial Preload	
p_rst_l	Reset	Reset	Reset	Reset	Reset	Performed	Release of **p_rst_l**
Chip reset bit[b]	Reset	Reset	Reset	Reset	Reset	Performed	Automatically after approximately 100 µs
D3 to D0 state transition[c]	Reset	Reset	Reset	Reset	Reset	Performed	Automatically after approximately 100 µs
Secondary reset bit[d]	No effect	Reset	No effect	Reset	Reset	Not performed	Secondary Reset bit
s_rst_in_l	Reset	Reset	Reset	Reset	Reset	Performed	Release of s_rst_in_l

a. **s_rst_l** is deasserted on reset release.
b. Reset control register, configuration offset D8h, bit [1].
c. Power management CSR, configuration offset E0h, bits [1:0].
d. Reset control register, configuration offset D8h, bit [0].

With the exception of assertion of the Secondary Reset bit and **s_rst_in_l**, all the other reset mechanisms cause a chip reset to occur and initiate a serial preload upon reset completion.

When the 21555 is reset by the assertion of **p_rst_l**, **s_rst_in_l**, or chip reset, various modes are selected by means of pull-up or pull-down resistors on the **pr_ad[7:3]** signal pins, as listed in Table 2.

Table 2. Mode/Feature Selection During Reset

Mode/Feature	Pin	Pull-up	Pull-down	Notes
Secondary bus arbiter	**pr_ad[7]**	Enabled	Disabled	—
Secondary central function enable	**pr_ad[6]**	Disabled	Enabled	Enables **s_req64_l** assertion and driving **s_ad[31:0]**, **s_cbe_l[3:0]**, and **s_par** low during secondary reset.
s_clk_o enable	**pr_ad[5]**	Enabled	Disabled	**s_clk_o** is driven low when disabled.
Synchronous mode enable	**pr_ad[4]**	Disabled	Enabled	Only enable when **s_clk** is derived from **s_clk_o** to save a cycle of latency.
Primary lockout reset value	**pr_ad[3]**	Enabled (reset to 1)	Disabled (reset to 0)	When the Primary Lockout bit is 1, primary configuration accesses receive target retry.
Serial ROM preload	**pr_ad[2]**	Disabled	Disabled	Setting the pin to either high or low to disables this feature.
Primary bus 64-bit extension	**pr_ad[1]**	Disabled	Enabled	If the s_rst_in_l signal is used to reset the chip, a low setting enables this feature.

3.2 Serial Preload

Whenever the 21555 configuration registers are reset either through assertion of p_rst_l, s_rst_in_l, setting the chip reset bit, or after a power power management transition from D3hot to D0. Once reset is complete by detecting p_rst_l and s_rst_l deasserts and the chip reset bit to 0, it initiates a serial ROM read in order to perform a configuration register preload. This preload is used to:

- Select the size and type of downstream, upstream, and ROM address windows by pre-loading address setup configuration registers
- Load read-only configuration values such as subsystem vendor ID, subsystem ID, class code, MAX_LAT, MIN_GNT
- Enable device specific features such as, read and write queue tuning thresholds, additional error modes, I_2O enable, and so forth in the chip control 0 and 1 configuration registers
- Configure the bus master priority levels of the secondary bus arbiter
- Enable/disable SERR# for specific error conditions
- Set up the power management register interface for the subsystem, including supported power states, and PME# enable
- Clear or set the Primary Lockout bit in the chip control register, which will enable or retry primary configuration accesses, respectively

Serial preload is not used to initialize those registers normally configured by initialization code such as the command register, base addresses, cache line size, and master latency timers.

While the serial preload is taking place, all configuration register accesses to the 21555 receive a target retry (the serial ROM preload takes approximately 570 µs). Once serial preload is complete, the local and host processors must perform secondary and primary configuration and initialize translated base registers.

Refer to the *Using a Serial ROM with the 21555 Non-Transparent PCI-to-PCI Bridge Application Note* and *Using a Flash ROM with the 21555 Non-Transparent PCI-to-PCI Bridge Application Note* for additional information on the ROM interfaces, programming, and sample register preload sequences.

3.3 Local Configuration

Under the typical initialization scenario shown in Figure 1, serial preload sets the Primary Lockout bit, which allows the local processor to have access to 21555 configuration space before the host processor. During primary lockout, the 21555 will retry accesses by the host processor on the primary bus. This allows the local processor to override serial preload state, if desired, and to perform additional initialization before the host processor configures the 21555 primary registers.

The local processor may set up the following 21555 state:

- Translated base addresses corresponding to upstream and downstream address windows
- Secondary bus configuration state, such as secondary bus base address, cache line size, master latency timer, and Command register enables
- The Upstream Memory 2 Look-up Table
- I_2O pointer and counter initialization

After the local processor has completed configuration of the 21555, it must clear the Primary Lockout bit in the chip control register to allow host processor access to the 21555. Care must be taken not to exceed the maximum time allowed before host access to configuration registers. The *PCI Local Bus Specification, Revision 2.2* specifies that PCI configuration space must be available for access by the host within 2^{25} primary PCI clock cycles.

Although the local processor also has access to primary bus configuration registers, initialization of these registers is typically done by the host processor.

3.4 Host Configuration

After the local processor completes initialization and clears the Primary Lockout bit, the host may then access 21555 configuration space. Although the host processor has access to most secondary bus and device-specific registers, typically the host processor only accesses the primary bus PCI configuration registers to perform plug and play initialization, such as:

- Primary base address registers
- Primary master latency timer
- Primary cache line size
- Primary interrupt line
- Primary command register enables

4.0 Allocating Address Ranges

The 21555 address registers fall into the following categories:

- Base address registers mapping 21555 CSRs.

- Setup registers specifying the type and size of downstream, upstream, and expansion ROM BARs.
- Base address registers mapping address ranges for downstream and upstream forwarding.
- Base address register mapping expansion ROM (on the primary interface only).
- Translated base registers for address translation of downstream and upstream transactions.
- Look-up table and associated registers for support of the Upstream Memory 2 range.

The 21555 always requests memory and I/O space on both buses to map its CSRs. Access to 21555 CSRs does not require setup registers or translated base address registers.

4.1 Address Range Setup

To specify an address range for downstream or upstream forwarding, the setup register corresponding to the BAR must be initialized[a]. This can be done either during SROM preload or from the local processor, and must be performed before the Base Address registers are mapped.

The setup register should not be changed once the corresponding BAR has been mapped. The host processor cannot access the setup registers. The reset value of the setup registers disables the corresponding BAR for transaction forwarding—the BAR will not request address space.

Writing a 1 to the most significant bit of the setup register enables the BAR; if this bit is equal to 0 the BAR is disabled and requests no space.

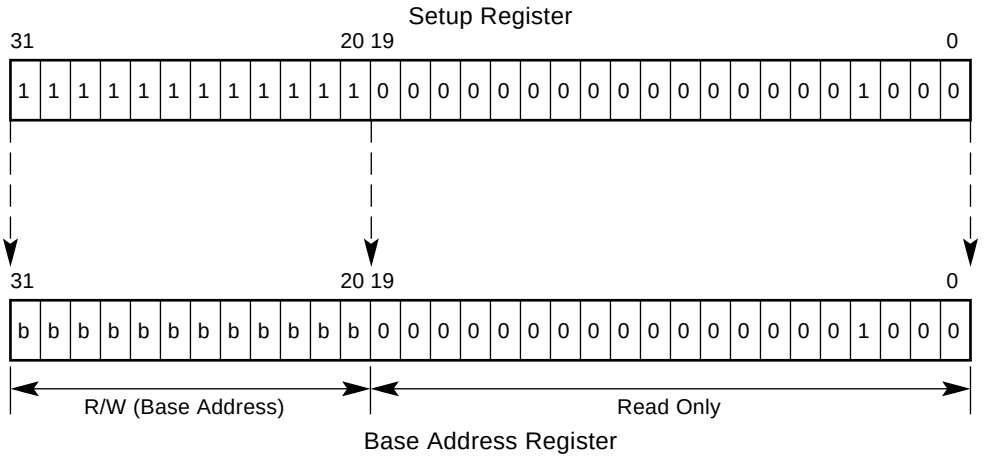
Bit 0 of the Downstream I/O or Memory 1 Setup register and the Upstream I/O or Memory 0 Setup register should be written with a 0 to select a memory BAR or a 1 to select an I/O BAR. Bits [2:1] select the type of memory mapping. The Downstream Memory 3 Setup register bits [2:1] may be set to 10b to select 64 bit addressing.

A mask is used to set the size of the BAR for the remaining read/write bits of the setup register. Writing a 1 sets the corresponding bit in that setup register's BAR to be read/write. Writing a 0 sets the corresponding bit in that setup register's BAR to be read only as 0. Therefore, the size is established by writing the appropriate number of most significant bits to a 1, and the remaining bits to a 0. This is consistent with the requirement that a PCI address range defined by a BAR must be a power of 2 in size, aligned to that same power of 2 (for example, a 1 MB address range aligned to a 1 MB boundary).

Figure 2 shows an example of using a setup register to program a BAR to request 1MB of memory space.

a. The size of the Upstream Memory 2 address range is specified by setting the page size in the chip control 1 configuration register; it does not have a setup register.

Figure 2. Setup Register Example



4.2 Setup Initialization Code Example

The following code example shows a typical setup sequence where the local CPU scans for the 21555 configuration space on the secondary side of 21555. When found and identified, the configuration space information (idsel) is stored for later use. The local CPU then proceeds to set up the 21555 BAR sizes by writing to the specific setup register for each BAR. If an serial ROM is present, this code will overwrite the contents loaded into these setup registers from the serial ROM. The init_complete (primary lockout) feature of the 21555 guarantees that the host will not access any of the BARs from the primary side until all setup operations are complete.

```
/* local CPU scans PCI bus for 21555 (secondary side) */
/* when 21555 found, save configuration information into idsel var */
for (ii=0; ii < 32 && !done ; ii++ ) {
   if (pciConfigReadDword(ii, vendorIdOffset) == 21555_device) {
      done = true;
      idsel = ii;
   }
}

/* Initialize downstream setup registers on 21555 (from secondary side) */
/* Primary downstream bar0 requests 8MB prefetchable memory, page 6-37 */
pciConfigWriteDword( idsel, 0xAC, 0xFF800008 );

/* primary downstream bar1 requests 256 bytes IO space, page 6-35 */
pciConfigWriteDword( idsel, 0xB0, 0xFFFFFF01);

/* primary downstream bar2 requests 8MB prefetchable memory, page 6-37 */
pciConfigWriteDword( idsel, 0xB4, 0xFF800008);

/* primary downstream bar3 request disabled, page 6-37 */
pciConfigWriteDword( idsel, 0xB8, 0x0);
```

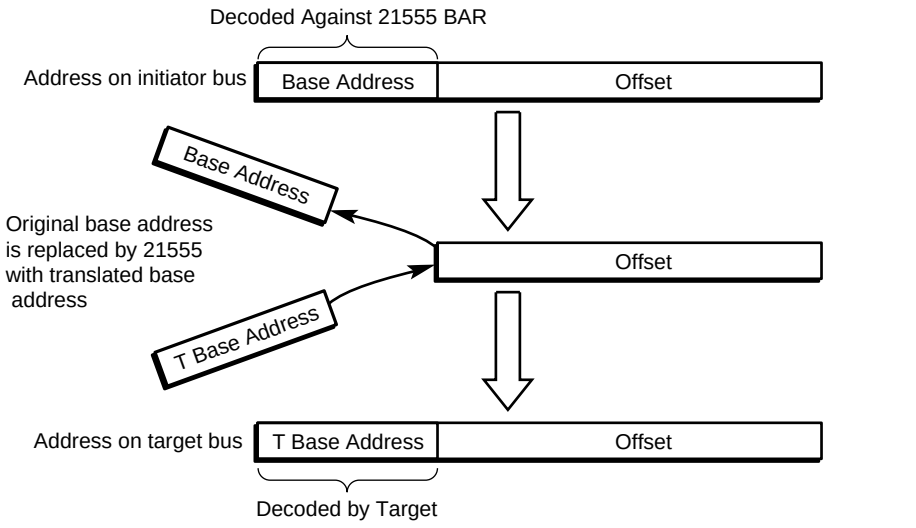
```
/* primary downstream bar3 upper32 bits request disabled, page 6-38 */
pciConfigWriteDword( idsel, 0xBC, 0x0);

/* initialize up stream setup registers on 21555 (from secondary side) */
/* secondary upstream bar0 requests 256 bytes IO space, page 6-35 */
pciConfigWriteDword( idsel, 0xC4, 0xFFFFFF01);

/* secondary upstream bar1 requests 8 MB prefetchable memory space, page 6-37 */
pciConfigWriteDword( idsel, 0xC8, 0xFF800008);
```

4.3 Mapping Address Ranges

Once the setup registers have been configured, then the corresponding BARs may be mapped. The BARs are the standard PCI registers that request PCI address resources. The downstream BARs are mapped in primary PCI address space and the upstream BARs are mapped in secondary PCI address space.

Before any memory or I/O transactions are forwarded across the 21555, the translated base registers must be configured. The translated base register specifies the base address of the window on the target (opposite) bus. When the transaction is forwarded, the translated base address is substituted for the original base address as shown in Figure 3. The translated base registers do not affect the mapping of the BARs, and may be initialized before or after BAR mapping, but before the BAR is used to decode transactions.

Figure 3. Address Translation Using Translated Base Address



4.4 Setting Translated Base Code Example

The following code example uses the idsel information gathered in the previous example. At this point, the code does not care about where the host has mapped the BARs on the primary side of the 21555. The code does care about where those BARs map to on the secondary (local) side. So the code instructs the 21555 to translate addresses decoded by the 21555's primary side to somewhere pertinent on the secondary side (i.e. local memory) by writing the translated base registers associated with each BAR. Upstream translation allows local CPU software to use local fixed addresses to access host-allocated buffers or devices anywhere in the host domain address space.

```
/* All dowstream transactions need to point to valid local domain locations */
/* downstream translated base, page 6-34 */
pciConfigWriteDword( idsel, 0x94, BAR0 PRI TO SEC XLATE );
pciConfigWriteDword( idsel, 0x98, BAR1 PRI TO SEC XLATE );
pciConfigWriteDword( idsel, 0x9c, BAR2 PRI TO SEC XLATE );
pciConfigWriteDword( idsel, 0xA0, BAR3 PRI TO SEC XLATE );

/* All upstream transactions need to point to valid host domain locations */
/* upstream translated base, page 6-33 */
/* this is typically negotiated with host driver at runtime */
pciConfigWriteDword( idsel, 0xA4, BAR0 SEC TO PRI XLATE );
pciConfigWriteDword( idsel, 0xA8, BAR1 SEC TO PRI XLATE );
```

4.5 Mapping BARs Code Example

The following code example shows the local CPU mapping the 21555 into the local domain memory and I/O address space. This will typically be done for all of the PCI devices in the local domain using a PCI configuration manager running on the local CPU.

```
/* now configure 21555 BARs. Will typically happen in local PCI plug&play module*/
/* set secondary command register to 0 */
pciConfigWriteWord( idsel, 0x04, 0);

/* set upstream BARs to local domain addresses */
pciConfigWriteDword( idsel, 0x10, LOCAL DOMAIN MEM ADDR);
pciConfigWriteDword( idsel, 0x14, LOCAL DOMAIN IO ADDR);
pciConfigWriteDword( idsel, 0x18, LOCAL DOMAIN MEM ADDR);
pciConfigWriteDword( idsel, 0x1c, LOCAL DOMAIN MEM ADDR);

/* with all other secondary PCI devices. set master latency timer, cache line size
*/
pciConfigWriteWord( idsel, 0x0c, 0x4008);

/* set secondary command register, master, mem & IO, MWI, PERR, SERR enable */
pciConfigWriteWord( idsel, 0x04, 0x157);
```

5.0 Minimum Initialization Requirements

Table 3 lists the minimum setup required to enable the 21555 to forward memory transactions downstream and upstream. It also shows what mechanism typically initializes the register. This section does not describe initialization of the Upstream Memory 2 address range, which uses the look-up table.

Table 3. Register Initialization Required for Downstream and Upstream

Register	Preload	Local µP	Host µP	Notes
Downstream memory *x* setup	Yes[a]	Yes	**X**[b]	One or more ranges; *x* = 0,1,2,3
Upstream memory *y* setup	Yes	Yes	**X**	One or more ranges;*y* = 0,1
Downstream memory *x* translated base	[c]	Yes		—
Upstream memory *y* translated base		Yes		—
Upstream memory *y* base address		Yes		Standard initialization code
Secondary cache line size		Yes		Used for prefetching, MWI, and buffering guidelines
Secondary master latency timer		Yes		Non-zero to guarantee longer bursts
Secondary command		Yes		Memory and master enables Optionally: MWI enable, fast back-to-back enable, parity enable, SERR# enable
Downstream memory *x* base address			Yes	Standard initialization code
Primary cache line size			Yes	Used for prefetching, MWI, and buffering guidelines
Primary master latency timer			Yes	Non-zero to guarantee longer bursts
Primary command			Yes	Memory and master enables Optionally: MWI enable, fast back-to-back enable, parity enable, SERR# enable

a. Yes indicates that this is the expected or typical access method for initialization.
b. **X** indicates that there is no access to this register using this access mechanism.
c. A blank box indicates that access is allowed, but this isn't expected or typical access mechanism for initialization.